Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Previously Presented) A computer system having a data processing environment in which a program is divided into and executed as multiple threads, and in which said threads share and access data that is stored in a memory device, comprising:

means for indicating specific data that will be accessed only by a specific thread;

means for determining, when a thread attempts to access data, whether a specific thread indication is present relative to the data being accessed;

means for accessing said specific data without first performing a locking process to reject access attempts by other threads, when the specific thread indication is present; and

means for performing a locking process for the data being accessed before accessing the data when it is determined that no specific thread indication is present.

2. (Previously Presented) The computer system according to Claim 1, wherein said specific thread detects data, included in said data stored in said memory device, and said specific thread does not have a reference pointer to said data, and thereafter releases memory occupied by said data to provide storage space that is freely available.

3. (Previously Presented) In a data processing environment, a system in which multiple threads share and access objects, comprising:

flag data, provided for an object, for indicating an existence of a locality specifying that said object is to be accessed only by a specific thread;

means for having the specific thread access said object when said flag data for said object indicates said locality for said specific thread, without performing a locking process to reject access attempts by other threads or other objects before accessing said specific data; and

means for having the specific thread perform said locking process before accessing said object when said flag data does not indicate said locality for said specific thread.

4. (Previously Presented) The computer system according to Claim 3, wherein, when said object is created by a thread, said object sets said flag data indicating a locality exists for said thread, and wherein, before said object is changed so that it can be accessed by another thread or another object, said locality indicated by said flag data is canceled.

5. (Previously Presented) The computer system according to Claim 3, wherein said specific thread detects an object for which said flag data indicates the existence of a locality for said specific thread but said specific thread does not have a reference pointer to said data, and thereafter releases said object to provide in a memory device storage space that is freely available.

6. (Previously Presented) A memory management method for a data processing environment in which a program is divided into and executed as multiple threads, and in which said threads share and access objects that are stored in a memory device, comprising the steps of:

setting flag data indicating an existence of a locality for a specific object that is created by a specific thread and that is to be accessed only by said specific thread;

canceling said locality indicated by said flag data before said specific object is changed so that said specific object can be accessed by another thread;

without performing a locking process to reject access attempts by other threads or objects, accessing said specific object when said flag data for said specific object indicates the existence of a locality for said specific thread; and

locking said specific object before accessing said specific object when there is no said flag data indicating a locality for said specific thread.

7. (Previously Presented) The memory management method according to Claim 6, wherein said step of canceling said locality indicated by said flag data for said specific object includes a step of:

performing said locking process, when said specific object has a locality for a specific thread, that was skipped at the time said specific object was accessed by said specific thread.

8. (Currently Amended) A memory management method for a data processing environment in which a program is divided into and is executed as multiple threads, and in which said threads share and access objects that are stored in a memory device, comprising the steps of:

setting flag data indicating an existence of a locality indicating that a specific object that is created by a specific thread is to be accessed only by said specific thread;

permitting said specific thread to detect an object for which the flag data indicates the existence of a locality for said specific thread and said specific thread does not have a reference pointer to said object; and

releasing said detected object to provide additional storage space in the memory device that may be freely used.

9. (Currently Amended) Computer readable code stored on computer readable medium or permitting a locking step to be skipped in certain situations relative to data in a multi-thread environment, comprising:

[first subprocesses] a process for setting flag data indicating the existence of a locality for a specific object that is created by a specific thread and that is to be accessed only by said specific thread;

[second subprocesses] a process for canceling said locality indicated by said flag data before said specific object is changed so that said specific object can be accessed by another thread;

[third subprocesses] a process accessing said specific object when said flag data for said specific object indicates the existence of a locality for said specific thread without performing a locking process to reject access attempts by other threads; and

[forth subprocesses] a process for performing said locking process before accessing said specific object when said flag data indicates the absence of a locality for said specific thread.

10. (Currently Amended) Computer readable code stored on computer readable medium for performing memory management for a program that executes in multiple threads, comprising:

[first subprocesses] a process for setting flag data indicating existence of a locality

indicating that a specific object that is created by a specific thread is to be accessed only by said specific thread;

[second subprocesses] a process for permitting said specific thread to detect an object for which flag data indicates the existence of a locality for said specific thread and said specific thread does not have a reference pointer to said object; and

[third subprocesses] a process for unlocking said detected object so that storage space may be freely used.